

Satellite and Terrestrial Network Analysis

The Performance of Satellite Spoofing

Joseph Ishac
jai5@po.cwru.edu

Department of Electrical Engineering and Computer Science
Case Western Reserve University

Technical Report

January 10, 2001

Abstract

In this paper, we analyze the performance of protocols in a network that consists of both satellite and terrestrial components. One method, proposed by outside research, to improve the performance of network transfers over satellite is to use a performance enhancing proxy often dubbed spoofing. Spoofing involves the transparent splitting of a network connection between the source and destination by some entity within the network path. In order to analyze the impact of spoofing, we constructed a simulation suite based around the network simulator ns-2. The simulation reflects a host with a satellite connection to the Internet or terrestrial network and allows the option to spoof connections just prior to the satellite. The methodology used in our simulation allows us to analyze spoofing over a large range of file sizes and under congested conditions, while prior work on this topic has primarily focused on bulk transfers with no congestion. As a result of these simulations, we find that the performance gain of spoofing is less beneficial for smaller sized transfers than gains obtained when transferring large files.

1 Introduction

A growing topic in the past few years has been that of hybrid networks, or networks that contain both terrestrial and wireless links. While there are many forms of hybrid networks, the work presented in this paper focuses on the use of a geo-synchronous satellite within a network path. More specifically, the satellite link is located just prior to the user, similar to a DirectPC model. Although not discussed in this paper, satellites do carry some advantages over traditional cable. However, one of the main disadvantages of using a satellite in network communication is the long delay needed to transfer data to and from the satellite. Typical delays range on the order of half a second to travel from the ground to the satellite and back.

The Transmission Control Protocol (TCP) is the most widely used transport protocol for Internet traffic. One TCP feature in particular, congestion control, incorporates a slow-start mechanism, which is highly susceptible to high delay links [AKO00]. The end result is a decrease in initial performance, since it takes longer to build up the sending rate over a long-delay network path. Spoofing, which is discussed in more detail in Section 2, was introduced mainly for solving such a problem with high delays. However, prior work on spoofing has focused on simulations of bulk transfers without congestion, and has thus left an incomplete picture of spoofing's overall performance. The project discussed in this paper focuses on creating a simple yet versatile simulation environment, in which the performance of spoofing can be seen across a large range of file transfers, while under congestion conditions.

The remainder of the paper includes Section 2, which discusses spoofing in greater depth, and Section 3, which outlines the actual simulation mechanics and details. Section 4 presents the results of those simulations. Finally, Section 5 summarizes the conclusions and lists possible areas for future work on this subject.

2 Background

In an attempt to mitigate the disadvantages incorporated with long-latency links, researchers have been introducing performance-enhancing proxies (PEPs) into networks.

One such PEP that is currently being used in satellite networks is TCP spoofing [PILC-ID, I-TCP, ASBD96]. The objective of spoofing involves isolating the long-latency link by introducing a middle agent which splits the TCP connection (see Figure 1). However, unlike a proxy cache, spoofing is transparent to both the sender and receiver. Thus, the middle agent, or ‘spoofer’, takes on the personality of both parties. The responsibility of the spoofer is to intercept, cache, and acknowledge data received by the sender and then forward that data to the receiver. As a result, spoofing does break the end-to-end semantic of TCP, however while this raises several philosophical issues [PILC-ID], those issues are not the focus of this paper. Finally, it is worth noting that in our model data segments and connection teardowns are spoofed, while connection setup remains end-to-end.

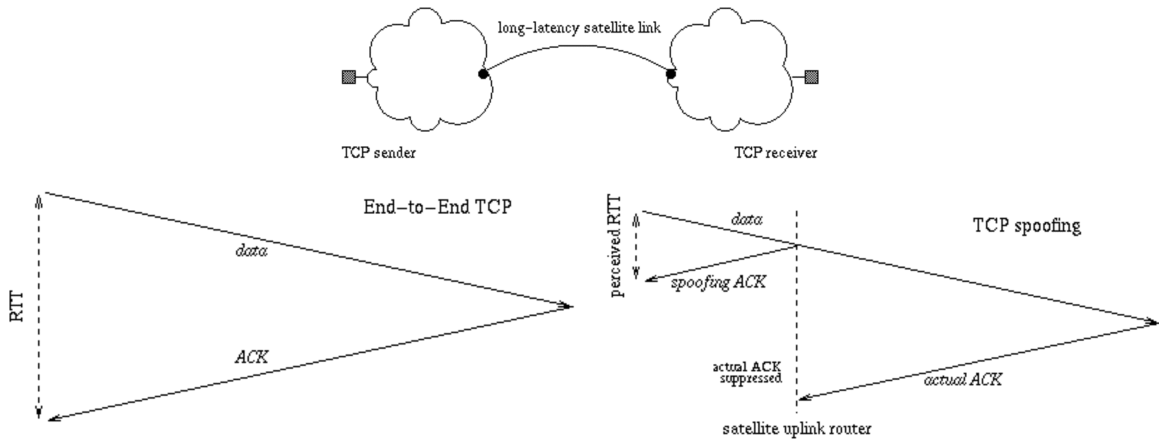


Figure 1: Satellite Spoofing

3 Simulation Overview

3.1 Terminology

The following terms are explained below and are used throughout this paper:

Flow: A series of transmissions from one host to another.

Packet: A TCP segment

For these simulations we used three metrics to measure the performance of a network flow. The first, throughput, is a measure of the time needed to complete a particular transfer and can be measured from either the sender or receivers perspective. For a sender side analysis, the time of completion is marked by the reception of the ACK for the final data packet. Whereas, the time of completion for receiver side analysis is marked upon transmission of the final ACK. The second metric used is goodput, which is an indication of what percentage of data packets were unique. Thus a goodput that is less than one would indicate a transfer that suffered from retransmissions. Since it is possible to have retransmissions without data loss (e.g. spurious timeout, lost ACK), the final metric used in the simulation was the calculation of the number of dropped data packets, or simply drops. The metrics are summarized in Table 1.

Throughput	$\frac{\text{Unique Packets}}{\text{Transfer Time}}$
Goodput	$\frac{\text{Unique Packets}}{\text{Total Packets}}$
Drops	Number of dropped data packets

Table 1: Summary of Metrics

3.2 Topology

The test network consists of five hosts and five routers as shown in Figure 2 below. Each host is connected to its appropriate router via an Ethernet link and runs TCP with selective acknowledgements (SACK) [RFC2018, FF96] and delayed ACKs [RFC1122, RFC2581]. Routers enforce drop-tail queuing on all links, but an option for enabling RED based queuing [FJ93] on both the satellite and Internet links was implemented. The thresholds for RED were based on suggestions from [Flo97] and are listed in the routing equations below. Finally segment sizes of 1500 bytes were used [All00].

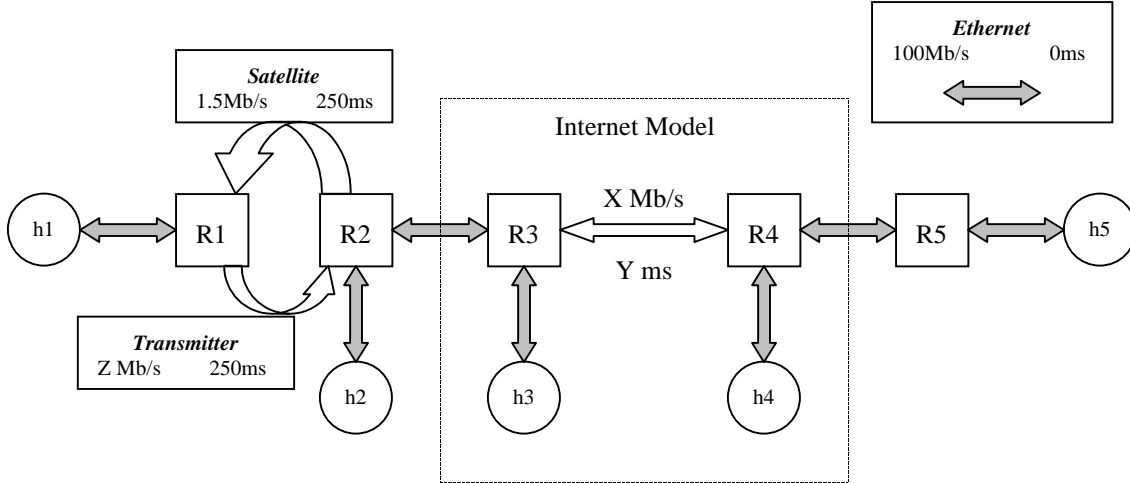


Figure 2: Network Topology

Queue Size

$$\text{Ethernet} = \infty$$

$$\text{Others} = (\text{Bandwidth in bytes per second} \cdot (2 \cdot \text{Delay})) / \text{Segment Size}$$

RED Queuing

$$\text{Threshold} = 1/5 \cdot \text{Queue Size}$$

$$\text{Maximum Threshold} = 3/5 \cdot \text{Queue Size}$$

(Only the integer part of the result is taken from all calculations)

Equation Set 1: Routing Calculations

The topology is laid out such that there are distinct satellite and Internet portions. First, the satellite is constructed asymmetrically to allow for the possibility of a low powered transmitter. Delay over the satellite is fixed at 250ms, and the download capacity set at a T1 rate. The capacity of the transmitter is specified at runtime. The second major portion of the topology, the Internet model, consists of four nodes. The link between the two routers acts as the “Internet”, whose bandwidth and delay are also

specified at runtime. The two hosts in the model are responsible for generating cross traffic over the Internet link. Also, even though the model for the Internet is simple and unrealistic of its real life counterpart, it is sufficient in capturing the basic characteristics of propagation delay, limiting bandwidth, and competing traffic.

The remaining hosts are the nodes at which connections of interest will take place. Using these three hosts, any combination of the hybrid network can be analyzed. More specifically, host one represents a pure satellite user, and host five represents a user with a high-speed connection to the Internet. Host two stands at a middle ground with access to both portions of the network. This property also makes host two capable of spoofing connections as shown in Figure 3.

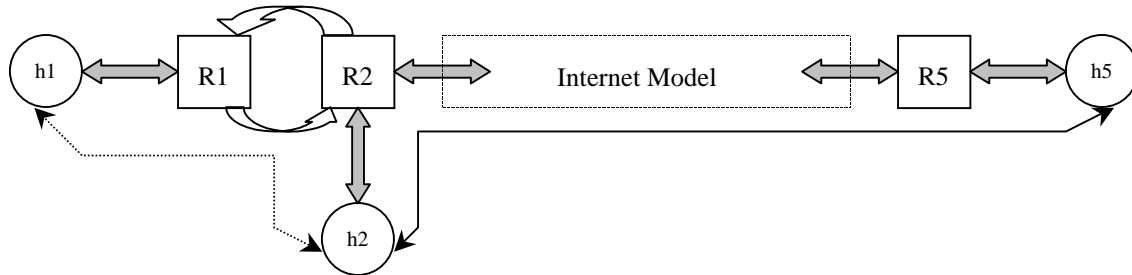


Figure 3: Spoofing in the Simulator¹

Thus, with spoofing enabled, a connection from host five to one would be spoofed at host two. It would cache, with infinite capacity, data received from host five and forward the data to host one.

3.3 Traffic

All transfers used in the simulation make use of the File Transfer Protocol (FTP) to transmit data. No competing traffic is present on the satellite link as it represents a dedicated satellite channel². However, competing traffic is present on the Internet link by using an analytical FTP generator which is discussed in detail in [Ish01]. Traffic can also be generated by using a trace file to recreate sessions observed on a real network.

3.4 Software

The simulations in this paper make use of the *Network Simulator* (ns) [NS] version 2.1b6 with two bug fixes, outlined in Appendix A. Traffic generation is separated from the simulation so as to facilitate reuse and modularization. The overall layout of software structure is shown in Figure 4. The output from ns consists of three trace files which are uniquely named in relation to the type of simulation being done. This allows the analyzer to distinguish which traces to analyze and also allows for congruent execution of simulations. The function of the controller is to synchronize the spawning and execution of both programs. Finally, a script automates the entire process, synchronizing traffic generation with the controller and allowing for multiple runs of different case scenarios.

¹ Due to simulator restrictions, spoofing could not be done at routers as would likely be the case if it were implemented in a real network. However, moving it to the host adds only the Ethernet delay which is neglect able and likely much smaller than any processing delays that would be present in a real system.

² The satellite in this simulation was based off of the Advanced Communications Technology Satellite (ACTS), which supported packet switching, spot transmissions, and frequency reuse.

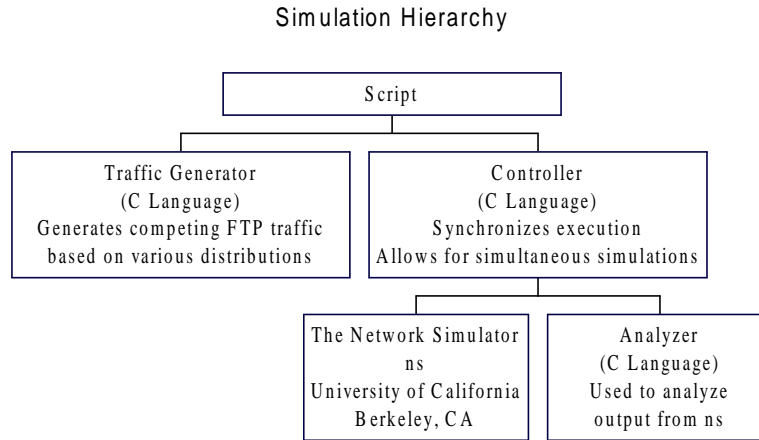


Figure 4: Software Layout

4 Results

The results detailed in the following section are based on a 30 run simulation with the following characteristics:

- T1 rate for Internet bandwidth: Note that while the Internet has a physically larger capacity than that of a T1, the allocation of bandwidth is regulated and so an actual obtainable value is less. Also, earlier test based on this simulation showed that changes to either the Internet bandwidth or delay have a predictable effect on the measured metrics. Therefore, in order to minimize the effect of extraneous bottlenecks, a T1 rate was used.
- An Internet delay of 0.069 seconds: This value came from sampling the delay of several sights at various geographical distances and averaging the result. Again, the delay does have an effect on the simulations. However, the purpose of these experiments was not focused on changes to the Internet properties and so the delay is sufficient in characterizing a typical value.
- T1 rate for transmission to the satellite: While this rate is much higher than what is economically feasible for a home user, the effect of variances to the transmission capacity was not of interest for this set of experiments.
- Transmission of files from the network user (host five) to the satellite user (host one) under the following granularity: 0 to 100 packets by 1, 110 to 500 packets by 10, 600 to 900 packets by 100, 1000 to 2000 packets by 1000
- Network variances: Drop-tail and RED queuing, end-to-end TCP connections and spoofing, sender side and receiver side analysis.
- For the sake of simplicity, end-to-end TCP is referred to as 'Regular' TCP in any subsequent plots.

Figures 5 and 7 show the throughput of the main flow ($h5 \rightarrow h1$) from either the receiver or sender side respectively. The different setup combinations between the queuing and PEP types represent the four curves on the plot. From the two plots we can see that, with our simulations, the use of RED based queuing has little effect on throughput in either case. Also, in the long run the steady state throughput values are nearly the same regardless of whose viewpoint is taken (as one would expect).

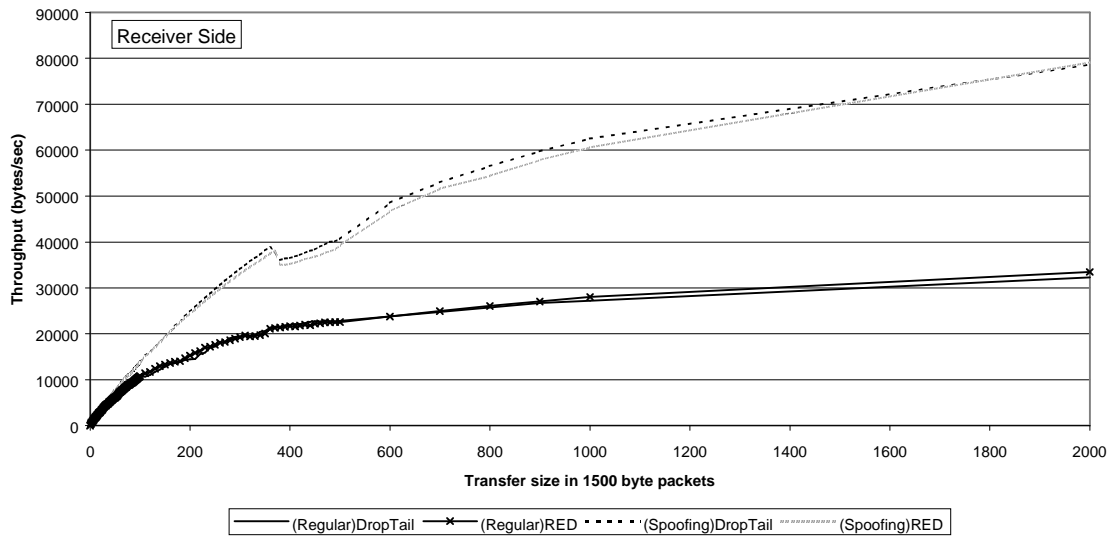


Figure 5: Throughput vs. Transfer Size (Receiver)

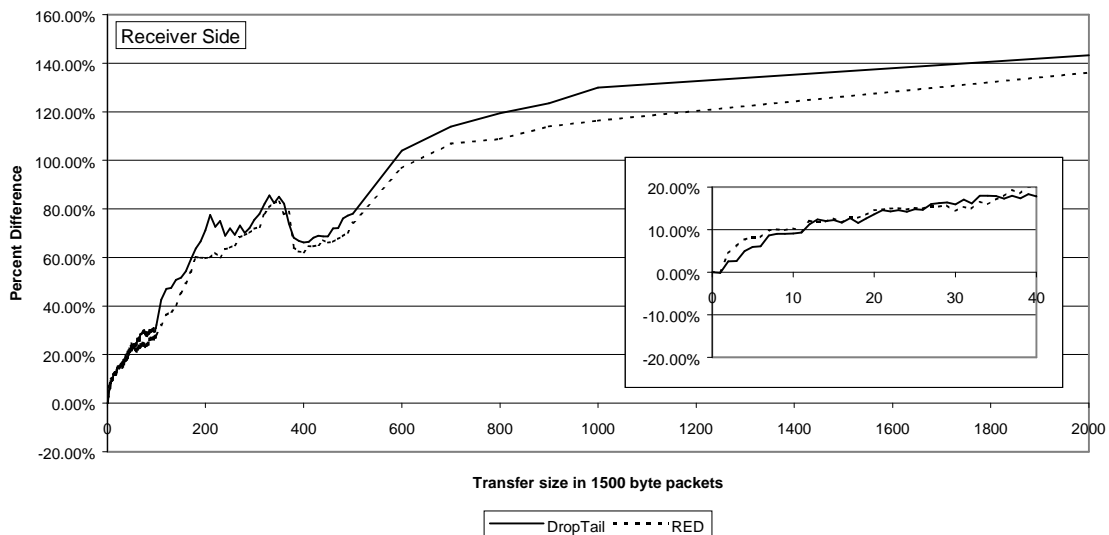


Figure 6: Percent Difference of Throughput (Receiver)

Figures 6 and 8 show the percent difference of the throughput that was shown in figures 5 and 7. The percent difference was calculated by taking the throughput difference of spoofing and end-to-end TCP over the throughput of end-to-end TCP. In general:

$$\text{Percent Difference} = (\text{PEP} - \text{Base}) / \text{Base},$$

Where the base in this case would be the throughput obtained from end-to-end TCP. Thus, the line identified as “DropTail” in the plot refers to the percent difference between “Regular-DropTail” and “Spoofing-DropTail”. For metrics, such as throughput, where a larger value indicates better performance, a positive percent difference indicates that the PEP outperformed the base condition, while a negative value indicates the exact opposite.

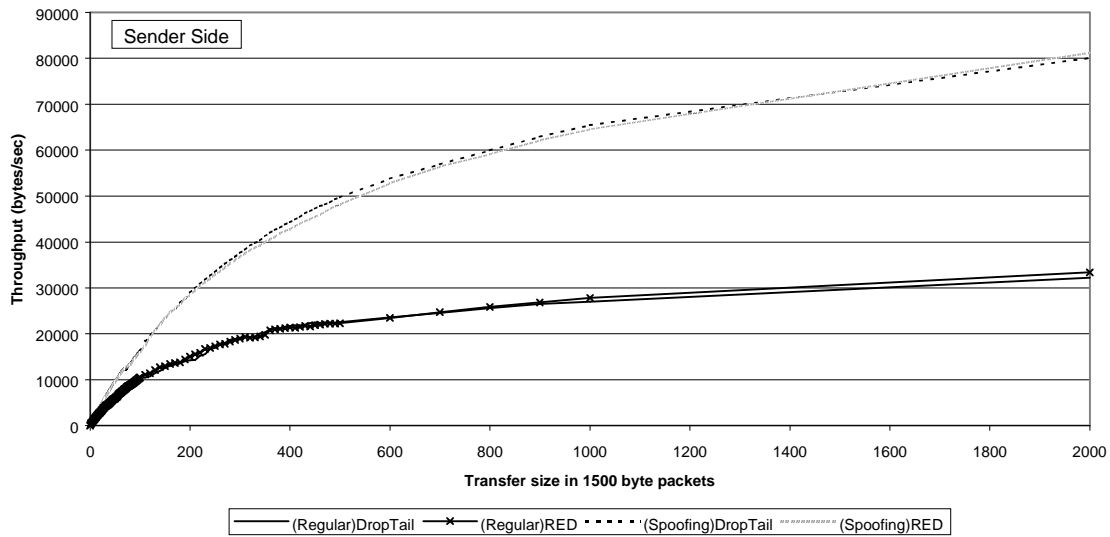


Figure 7: Throughput vs. Transfer Size (Sender)

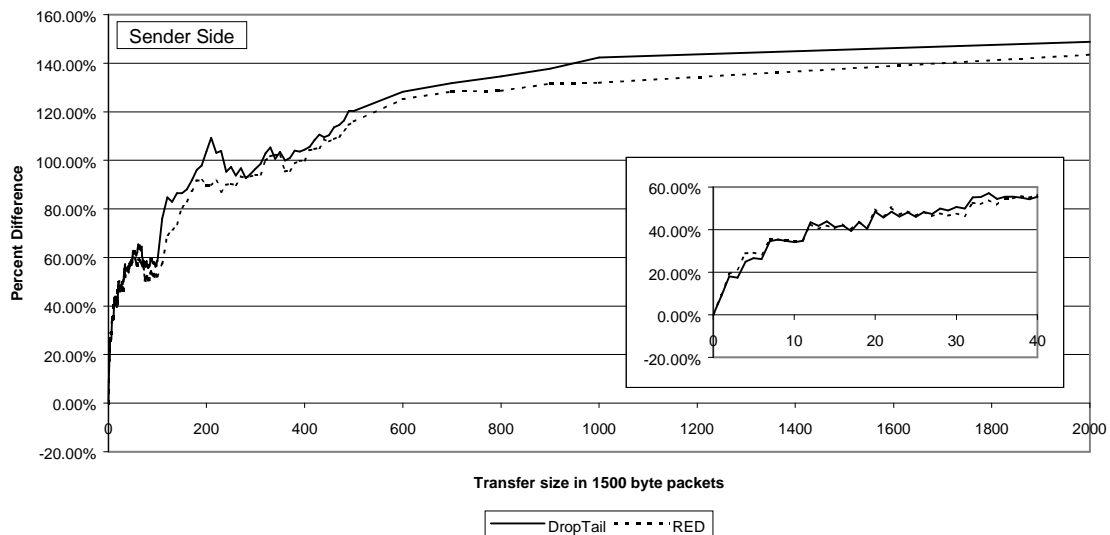


Figure 8: Percent Difference of Throughput (Sender)

Also included in the percent difference plot is an enlarged view of transfers consisting of 40 packets or less. The importance of considering small transfers is shown in Figure 9.³ From this plot, we see that transfers consisting of ten packets or less account for 90% of the network traffic. Also, although the data represents only a single network, the underlining concept has been generally noted in other networks as well.

Plots for goodput and drops were included in Appendix B so as to not clutter the main document. These plots follow the general principle of that seen with throughput, although both goodput and drops are independent of viewpoint. Also, unlike throughput and goodput, a positive percent difference in the number of drops indicates that the base condition outperforms the PEP since an increase in drops is unfavorable.

³ Cumulative Distribution Function (CDF) of network transfer sizes in packets as seen at the NASA GRC firewall on October 30, 2000.

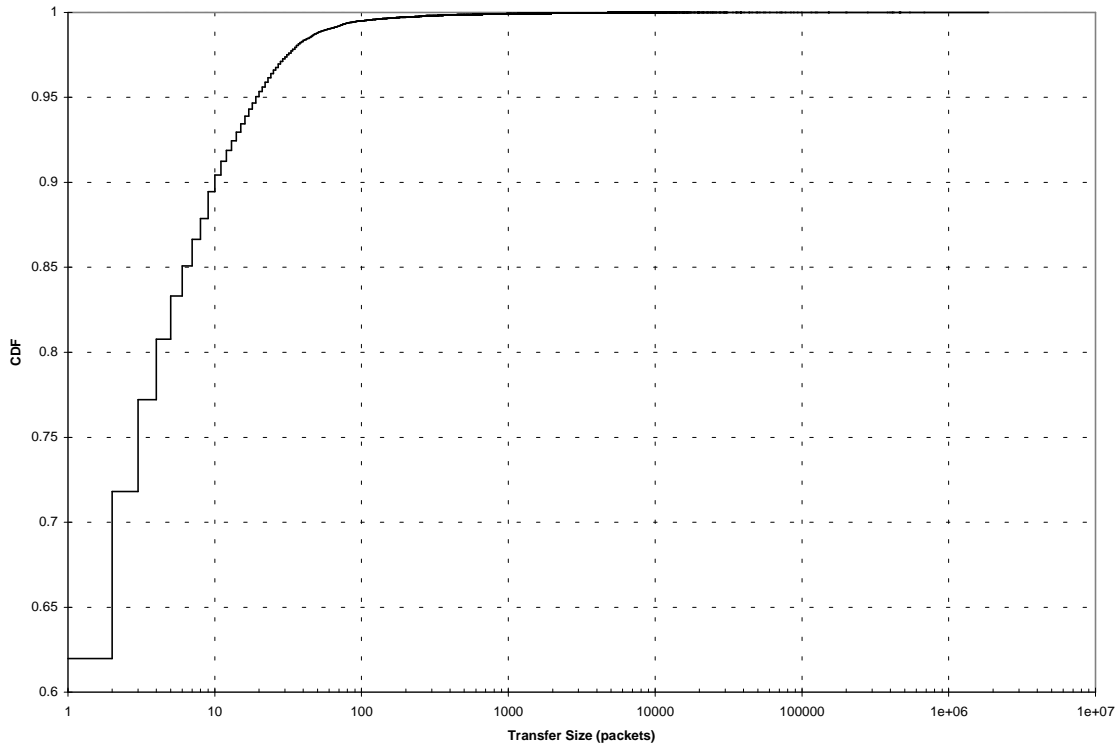


Figure 9: Distribution of Transfer Sizes

From the percent difference plots we can see that spoofing has very different effects depending on which vantage is used. From the receivers viewpoint, there is little effect for very small transfers and at most a 10% gain for transfers of ten packets or less. However, from the sender's perspective, the gain is much larger for those same small transfers. The large gain in throughput for the senders could be very beneficial for busy web servers. By freeing resources associated with long-delay connections quickly, it allows servers to satisfy more requests. Again, in the long run, both viewpoints show relatively the same performance gain. Both graphs contain some turbulence. Turbulence refers to several short and successive increases and declines, which can be attributed to the fact that the throughput values were an average of 30 runs and that the variance was very large. However, the dip in throughput found around 400 packets in Figure 6 is rather unusual. One possible cause is that the spoofer receives data more quickly, and thus overruns the satellite channel, dropping a large number of packets. This is possible since spoofing allows the rate of incoming packets to increase over the Internet and accumulate at the spoofer, which is still in early slow-start. Thus, spoofing adds a second bottleneck into the network path. The plot of drops shows a sharp increase around 400 packets when spoofing is used, which is supportive of this claim.

5 Conclusions and Future Work

As a result of our simulation, we found that spoofing is indeed beneficial for large file transfers. For small transfer sizes, spoofing greatly increased the throughput seen by the sender, but was much less beneficial for throughput observed at the receiver, which is the

vantage point perceived by the end-user. Since a majority of data sent across networks is small, spoofing will not provide much advantage to a standard home user. However, benefits to web servers and other content providers would be significant. Also, spoofing allows for data to accumulate at the spoofer, creating a second bottleneck and increasing the number of dropped data packets, which also degrades the receivers perceived performance.

We realize that the work done in this paper reflects simulations and not actual data measurements on real networks. Thus, a natural extension of this work may involve implementing these simulations in actual network test beds. Other extensions involve the inclusion of other types of PEPs as well as the effect of changes to the asymmetry of the satellite and to the characteristics of the Internet.

6 Acknowledgements

I would like to thank Mark Allman for providing the solution for the RTO bug within ns-2 and for the plot of transfer sizes at NASA. Also, I would like to thank both Mark Allman and Funda Ergun for commenting on earlier drafts of this report, and for continuous support and guidance with this work in general.

This work was supported by the NASA Glenn Research Center in conjunction with Case Western Reserve University under award number NAG3-2391.

References

- [AKO00] M. Allman, H. Kruse, and S. Ostermann, "A History of the Improvement of Internet Protocols Over Satellites", *Proceedings of the ACTS Conference 2000*, May 2000
- [All00] M. Allman, "A Web Server's View of the Transport Layer", *ACM Computer Communication Review*, 30(5), October 2000
- [ASBD96] Vivek Arara, Narin Suphasindhu, John S. Baras, and Douglas Dillon, "Asymmetric Internet Access Over Satellite-Terrestrial Networks", *Proceedings of the AIAA: 16th International Communications Satellite Systems Conference and Exhibit*, Part1, pp. 476-482, Washington, D.C., February 1996
- [FF96] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communications Review*, 26(3), pp. 5-21, July 1996
- [FJ93] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, 1(4), pp. 397-413, August 1993
- [Flo97] S. Floyd, "RED: Discussions of Setting Parameters", November 1997 <http://www.aciri.org/floyd/REDparameters.txt>
- [I-TCP] A. Bakre, B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 1995
- [Ish01] J. Ishac, "FTP Traffic Generator", Technical Report, Case Western Reserve University, January 2001

- [NS] VINT project: <http://www.isi.edu/nsnam/vint/index.html>
- [PILC-ID] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies", Internet-Draft, November 2000
draft-ietf-pilc-pep-05.txt
- [RFC1122] R. Braden, "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989
- [RFC2018] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996
- [RFC2581] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999

Appendix

A NS Bug Fixes

A.1 Delayed ACK Timeout

Problem:

This bug arises when resetting a connection that uses delayed acknowledgements. The ACK transmission timeout is not cancelled when a connection is reset. Also, resetting a connection clears any pending ACKs, which are saved globally. However, since the timeout is not cancelled correctly, upon expiration the simulator attempts to access a packet that no longer exists. The result is a fatal error and an ungraceful termination.

Solution:

A reset function for the delayed ACK sink is written to override the base class reset function (TCP sink). The new method correctly resets the ACK timer and calls upon its parent to finish the remaining portion of the reset routine. Also, several checks for null pointers were needed in other areas of the code.

A.2 Retransmission Timeout Calculation

Problem:

This bug arises when resetting a connection. The retransmission timeout (RTO) sampling index is not updated correctly when resetting a connection. The result is an incorrect sampling of path delay, which yields very large RTOs.

Solution:

Modify the reset method to correctly update the RTO sampling index to its initial value.

B Additional Results

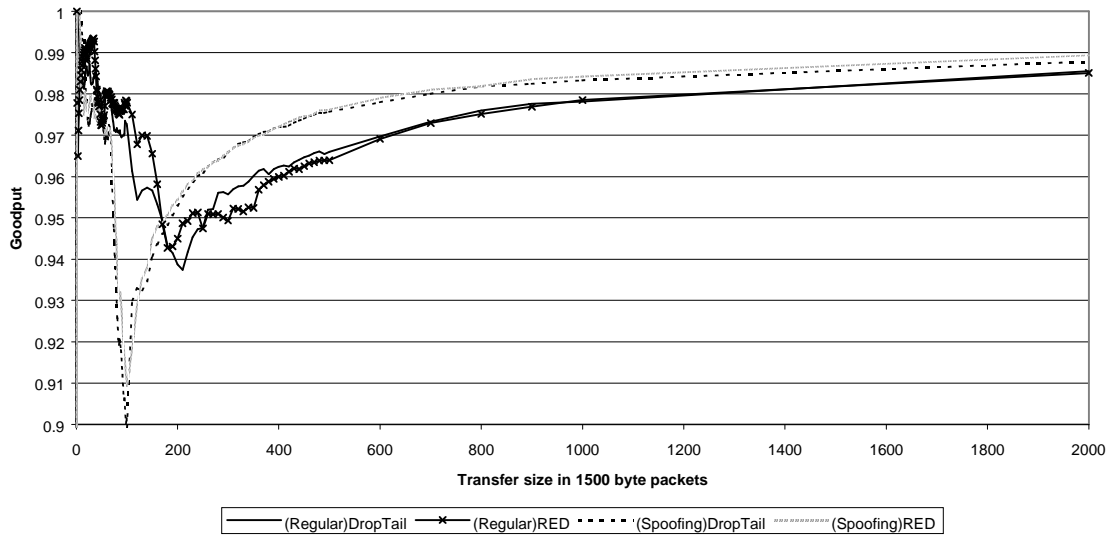


Figure 10: Goodput vs. Transfer Size

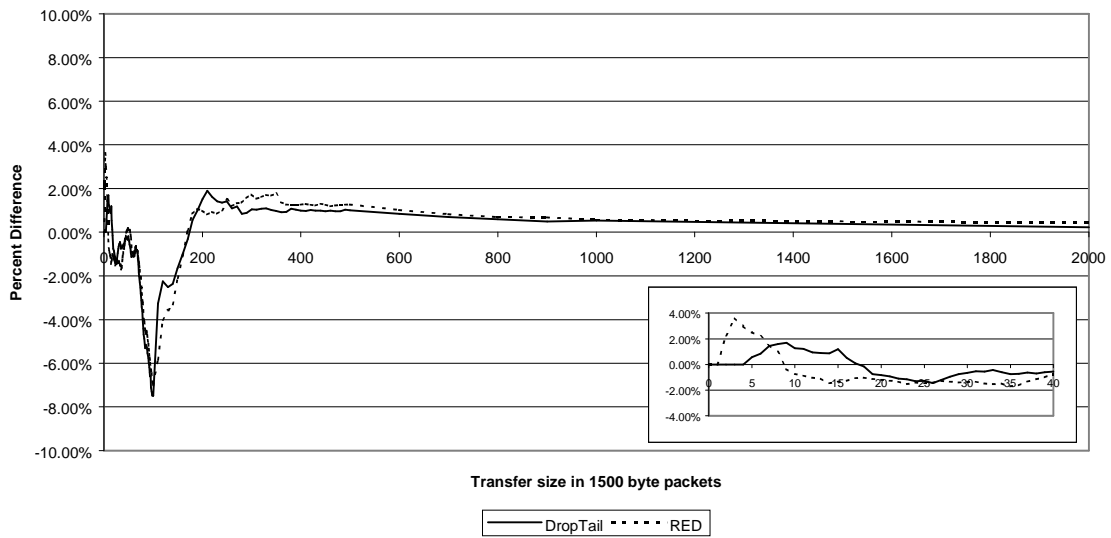


Figure 11: Percent Difference of Goodput

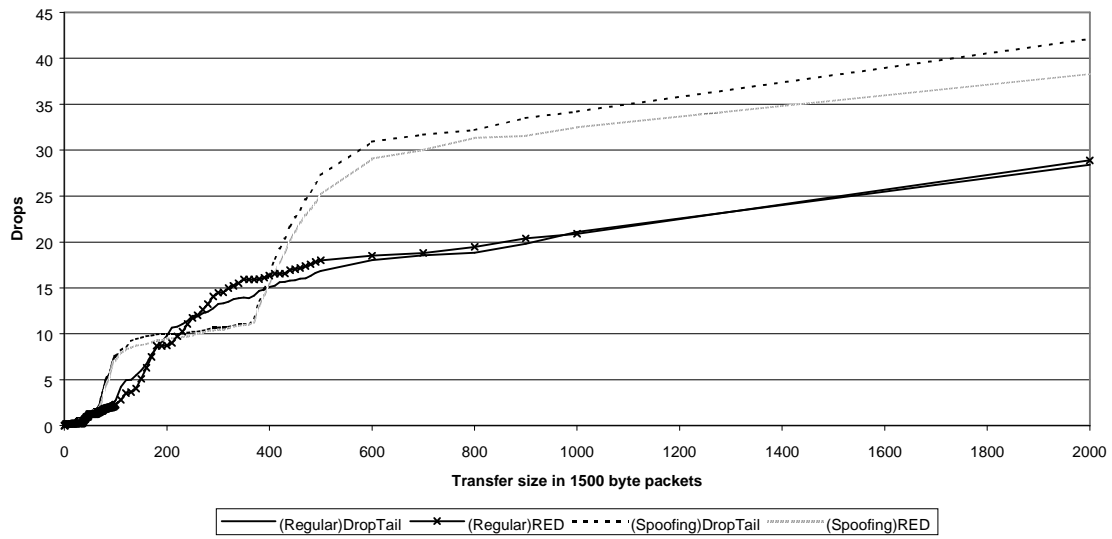


Figure 12: Number of Drops vs. Transfer Size

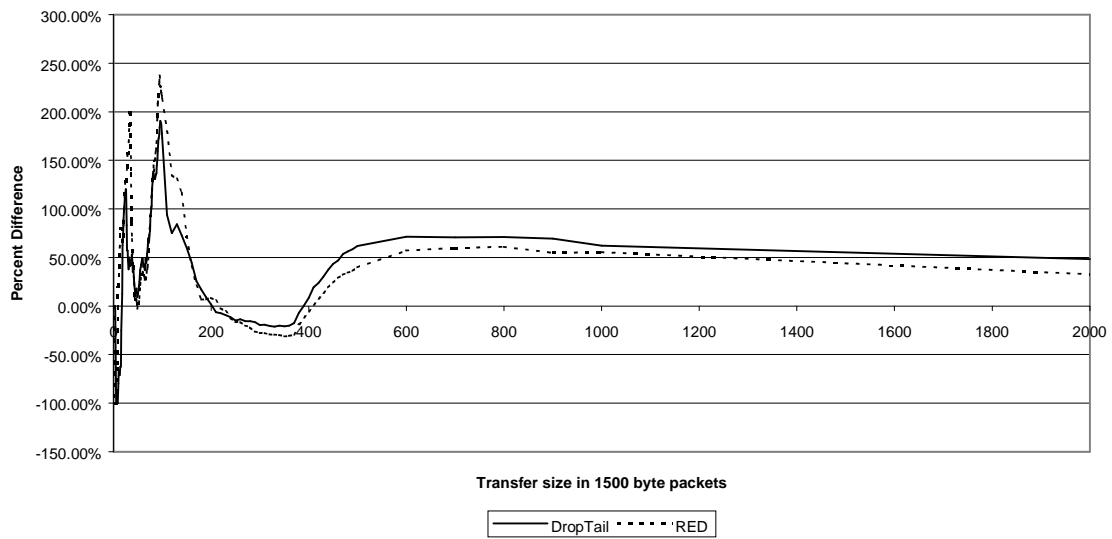


Figure 13: Percent Difference of Drops